What is claimed is:

1. A method, comprising:

creating one or more data structures sufficient to model an electronic circuit as a collection of n elements consisting of:

zero or more LRV elements, each having at least one of (a) a non-zero inductance parameter L_{br}, (b) a non-zero resistance parameter r_{br}, or (c) a non-zero voltage source parameter e_{br}, but neither a non-zero capacitance parameter, nor a non-zero current source parameter, nor a switch parameter;

zero or more CRI elements, each having at least one of (a) a non-zero capacitance parameter C_{br}, (b) a non-zero resistance parameter r_{br}, or (c) a non-zero current source parameter j_{br}, but neither a non-zero inductance parameter, nor a non-zero voltage source parameter, nor a switch parameter; and

zero or more switching elements, each having a switch state and neither a non-zero inductance parameter, a non-zero capacitance parameter, a non-zero resistance parameter, a non-zero voltage source parameter, nor a non-zero current source parameter; and

automatically generating a first set of state equations from said one or more data structures; and

simulating operation of the electronic circuit by application of said first set of state equations;

wherein n is at least two, and the collection comprises either

an LRV element for which at least two of L_{br}, r_{br}, or e_{br} are non-zero, or

a CRI element for which at least two of C_{br}, r_{br}, or j_{br} are non-zero.

2. The method of claim 1, wherein said simulating comprises producing state output data, the method further comprising:

modifying the parameters in said first set of state equations as a function of said state output data.

3. The method of claim 1, further comprising:

modifying the parameters in said first set of state equations based on a time-varying parameter of at least one element in said collection.

4. The method of claim 1, further comprising:

generating a second set of state equations from said one or more data structures upon the occurrence of a first topology change event.

5. The method of claim 4, wherein said generating said second set of state equations comprises modifying only the subset of said first set of state equations that depend on the one or more switching elements that have changed.

6. The method of claim 4, wherein each unique vector of switch states represents a topology of the overall circuit, and further comprising:

storing said first set of state equations in a cache;

after a second topology change event, determining whether a set of state equations in the cache represents the new topology;

if said determining is answered in the affirmative, using the set of state equations that represents the new topology to simulate operation of the circuit after the second topology change event; and

if said determining is answered in the negative, building a third set of state equations that represents the new topology, and using the third set of state equations to simulate operation of the circuit after the second topology change event.

7. The method of claim 6, further comprising:

storing said second set of state equations in a cache;

after a third topology change event, deciding whether a set of state equations in the cache represents the new topology;

if said deciding is concluded in the affirmative, using the set of state equations from the cache that represents the new topology to simulate operation of the circuit after the third topology change event; and

if said deciding is concluded in the negative, building a new set of state equations that represents the new topology, and using the new set of state equations to simulate operation of the circuit after the third topology change event.

8. A method, comprising:

creating one or more data structures that together store characteristics of a plurality of active branches B^{active} that make up a graph of nodes and branches that form a circuit, wherein B^{active} consists of

a set B^L of zero or more inductive branches, each having a non-zero inductive component but neither a capacitive component nor a variable switch state;

a set B^C of zero or more capacitive branches, each having a non-zero capacitive component but neither an inductive component nor a variable switch state; and

a set B^A of additional branches, each having neither an inductive component, nor a capacitive component;

partitioning B^{active} into a first branch set B_{tree}^{active} and a second branch set B_{link}^{active}, where the branches in B_{tree}^{active} form a spanning tree over B^{active}, giving priority in said partitioning to branches not in B^L over branches in B^L;

sub-partitioning B_{link}^{active} into a third branch set B_{link}^{L} and a fourth branch set B_{link}^{CA}, where $B_{link}^{L} = B_{link}^{active} \cap B^L$;

identifying a fifth branch set B^{CA} as the union of

B_{link}^{CA},

$B^C \cap B_{tree}^{active}$, and

those branches in B_{tree}^{active} that form a closed graph when combined with B_{link}^{CA};

partitioning B^{CA} into a sixth branch set $\tilde{B}_{tree}^{CA}$ and a seventh branch set $\tilde{B}_{link}^{CA}$, where the branches in $\tilde{B}_{tree}^{CA}$ form a spanning tree over B^{CA}, giving priority in said partitioning to branches in B^C over branches not in B^C;

identifying an eighth branch set $B_{tree}^{C} = \tilde{B}_{tree}^{CA} \cap B^C$;

selecting a set of state variables comprising:

for each branch of B_{link}^{L}, either the inductor current or inductor flux, and

for each branch of B^C_{tree}, either the capacitor voltage or capacitor charge; and

simulating a plurality of states of the circuit using the set of state variables.

9. The method of claim 8, wherein said partitioning steps each comprise an application of a weighted spanning tree algorithm.

10. The method of claim 9 wherein, for some positive numbers w_L and w_C:

for the partitioning of B^{active}, a minimum spanning tree algorithm is used with weight function $\omega_L(b_j) = \begin{cases} w_L \text{ if branch } b_j \in B^L \\ 0 \text{ otherwise} \end{cases}$; and

for the partitioning of B^{CA}, a maximum spanning tree algorithm is used with weight function $\omega_C(b_j) = \begin{cases} w_C \text{ if branch } b_j \in B^C \\ 0 \text{ otherwise} \end{cases}$.

11. A system, comprising a processor and a computer-readable medium in communication with said processor, said medium containing programming instructions executable by said processor to:

build state equations for a first topology of an electronic circuit having at least two switching elements, wherein each switching element has a switching state;

solve said state equations at time t_i to provide a state output vector, in which at least two elements control the switching states of the switching elements;

calculate the value of a switching variable as a function of the state output vector, wherein the value reflects whether the switching state of at least one of the switching elements is changing; and

if the value of the switching variable at time t_i indicates that at least one of the switching elements is changing, determine a second topology of the electronic circuit for time t_i^+ and obtain state equations for the second topology.

12. The system of claim 11, wherein:

said programming instructions comprise a state equation building module, a solver module for ordinary differential equations, and a switching logic module;

said building is performed by the state equation building module;

said solving and calculating are performed by the solver module; and

said determining is performed by the switching logic module.

13. The system of claim 12, wherein said obtaining is performed by said switching logic module.

14. The system of claim 12, wherein said obtaining is performed by said state equation building module.

15. The system of claim 12, wherein:

at a time t_j, at least two switching elements are each either rising-sensitive or falling-sensitive switches, wherein

rising-sensitive switches change switching state if and only if a controlling element of the state vector has passed from a negative value to a non-negative value; and

falling-sensitive switches change switching state if and only if a controlling element of the state vector has passed from a positive value to a non-positive value; and

the function is the arithmetic maximum of

a maximum of all elements of the state vector that control rising-sensitive switches, and

the negative of the minimum of all controlling elements of the state vector that control falling-sensitive switches.

16. A system for simulating electronic circuits, comprising a processor and a computer-readable medium in communication with said processor, said medium containing programming instructions executable by said processor to read element parameters and node connection information from a data stream comprising at least one switch type specification, the at least one switch type specification being selected from the group consisting of:

a unidirectional, unlatched switch;

a bidirectional, unlatched switch;

a unidirectional, latched switch; and

a bidirectional, latched switch; and

wherein said instructions are further executable by said processor automatically to calculate state equations for the circuit given the states of switches specified by said at least one switch type specification.